Neural Network

1. Terminology
2. Forward Propagation
3. Back Propagation
4. AND OR NOR example
5. Advantages & Disadvantages
6. CNN

Bayesian Analysis

1. Classical vs. Bayesian
2. Advantages & Disadvantages
3. Terminology
4. Coin example
5. Conjugate Function (Beta)

建议阅读:

CNN → https://zhuanlan.zhihu.com/p/42559190

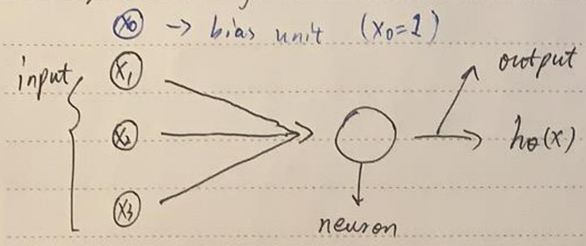
Back Propagation → https://www.cnblogs.com/charlotte77/p/5629865.html

Bayes 简单理解 → https://www.zhihu.com/question/21134457

Conjugate Prior → https://towardsdatascience.com/conjugate-prior-explained-75957dc80bfb

Neural Network 详细整理

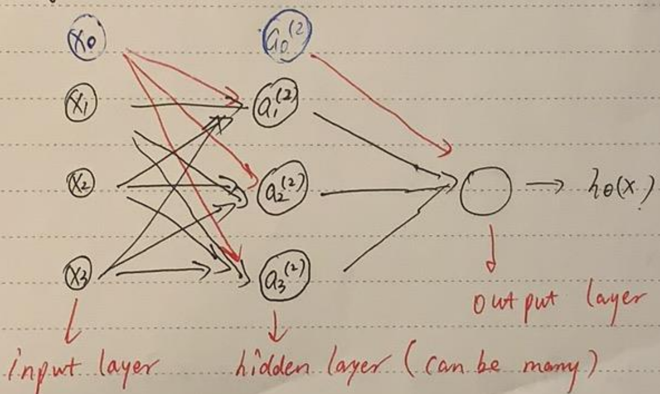
Origins : Algorithms to mimic the brain.



(logistic => $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$)

Here, sigmoid (logistic) function is also called activation function.

↓ make it complex



Where $a_i^{(j)}$ => "activation" of unit i in layer j

$\theta^{(j)}$ => matrix of weights controlling function mapping from layer j to layer $j+1$

So, we have

$$a_1^{(2)} = g(\theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\theta_{30}^{(1)} x_0 + \theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3)$$

$$h_\theta(x) = a_1^{(3)} = g(\theta_{10}^{(2)} a_0^{(2)} + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)})$$

If network has S_j units in layer j, S_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension $S_{j+1} \times (S_j + 1)$

Forward propagation : Vectorized implementation

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix} \qquad X \to a^{(1)}$$

$$\therefore z^{(2)} = \Theta^{(1)} a^{(1)}$$

$$a^{(2)} = g(z^{(2)}) \quad \to \mathbb{R}^3$$

$$\text{Add } a_0^{(2)} = 1 \quad \to a^{(2)} \in \mathbb{R}^4$$

$$z^{(3)} = \Theta^{(2)} a^{(2)}$$

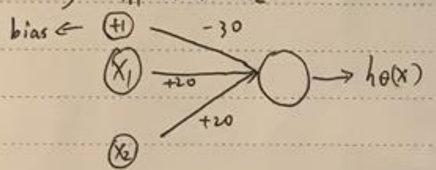
$$h_\Theta(x) = a^{(3)} = g(z^{(3)})$$

You don't need to understand the matrix math, just know how to initialize a NN.

☆ And example

$x_1, x_2 \in \{0, 1\}$

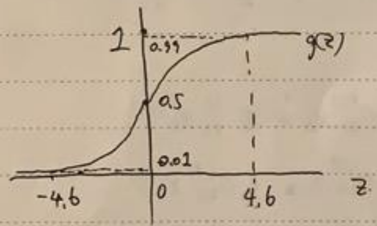
$y = x_1 \text{ AND } x_2$



assigning the weights for +1, x_1, x_2 => [-30, 20, 20]

So, $h_\Theta(x) = g(-30 + 20x_1 + 20x_2)$

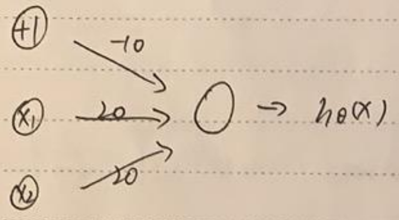
$$\Theta_{10}^{(1)} \qquad \Theta_{11}^{(1)} \qquad \Theta_{12}^{(1)}$$



x_1	x_2	$h_\Theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

$\therefore h_\Theta(x) \approx x_1 \text{ AND } x_2$.

OR example

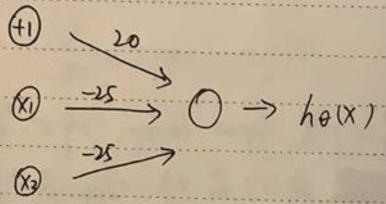


x_1	x_2	$h_\theta(x)$
0	0	$g(-10) \approx 0$
1	0	$g(10) \approx 1$
0	1	$g(10) \approx 1$
1	1	$g(30) \approx 1$

$\theta^{(1)} = [-10, 20, 20]$.

$\therefore\ g(-10 + 20x_1 + 20x_2) = h_\theta(x)$

NOR example



x_1	x_2	$h_\theta(x)$
0	0	$g(20) \approx 1$
1	0	$g(-5) \approx 0$
0	1	$g(-5) \approx 0$
1	1	$g(-30) \approx 0$

$g(20 - 25x_1 - 25x_2) = h_\theta(x)$

再回顾一下 forward propagation.

注意几点 ① The hidden layers no longer use the features (X's)

② They use linear combinations of the ~~code~~ nodes from the previous layer.

③ The outcome (Y's) are a function of the activations (Z's) in the last hidden layer.

$(z_1^{(1)} = \beta_{10}^{(2)} + \beta_{11}^{(2)} * a_1^{(2)} + \cdots)$

How to determine the weights : Back propagation.

步骤：
1. Set all weights to initial values that are random $\in (0,1)$
2. Calculate Y for ~~class~~ sample 1 using forward propagation
3. Calculate error
4. Adjust weights to minimize the error function using backward propagation
5. Repeat

So, let's talk about cost function (不考)

logistic :

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1-y^{(i)})(\log(1-h_\theta(x^{(i)})))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

NN :

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\theta(x^{(i)}))_k + (1-y_k^{(i)})\log(1-(h_\theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\theta_{ji}^{(l)})^2$$

Back propagation Algorithm

$\min_{\Theta} J(\Theta)$ Θ is matrix

$\frac{\partial}{\partial\Theta_{ij}^{(l)}} J(\Theta)$ => ? How to compute ?

So, remember what we did in the forward propagation :

$a^{(1)} = x$ (input layer)

$z^{(2)} = \Theta^{(1)} a^{(1)}$

$a^{(2)} = g(z^{(2)})$ (add bias unit $a_0^{(2)}$)

$z^{(3)} = \Theta^{(2)} a^{(2)}$

$a^{(3)} = g(z^{(3)})$ (add bias unit $a_0^{(3)}$)

Intuition : $\delta_j^{(l)}$ = "error" of node j in layer l.

For each output unit ($L=4$)

$\delta_j^{(4)} = a_j^{(4)} - y_j$ => output layer.

$a_j^{(4)} \longrightarrow (h_\theta(x))_j$

$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} .* g'(z^{(3)}) \longrightarrow a^{(3)} .* (1-a^{(3)})$

$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} .* g'(z^{(2)})$

Training set $\{(x^{(1)}, y^{(1)}), \cdots, (x^{(m)}, y^{(m)})\}$

Set $\Delta_{ij}^{(l)} = 0$ (for all l, i, j)

For $i = 1$ to m

Set $a^{(1)} = x^{(i)}$

Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \cdots, L$

Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$

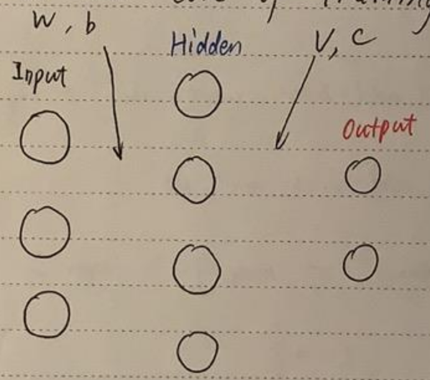
Compute $\delta^{(L-1)}, \delta^{(L-2)}, \cdots, \delta^{(2)}$

$\Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

$$\begin{cases} D_{ij}^{(l)} = \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} & \text{if } j \neq 0 \\ D_{ij}^{(l)} = \frac{1}{m} \Delta_{ij}^{(l)} & \text{if } j = 0 \end{cases}$$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$$

Backpropagation is the core of training a NN



W (D X M) ⇒ input weight matrix (to hidden)

b (M) ⇒ hidden bias

V (M × K) ⇒ hidden-to output weight matrix

C (K) ⇒ output bias

Goal is the same: build up $J(\theta)$, find gradients.

Another interpretation of the backpropagation
- Input → Hidden $\quad z = \sigma(w^T x + b)$ $\quad \sigma$ → activation function (we use Sigmoid by default)
- Hidden → output $\quad y = \text{Softmax}(V^T z + c)$ $\quad$ (For multiclassification, we use softmax.)
- Output → loss $\quad J = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log y_{nk}$ $\quad$ (cross entropy)

Note: we have many ways to define a loss function.

So, we use the chain rule:

$$\frac{\delta J}{\delta \beta^{(L)}} = \frac{\delta z^{(L)}}{\delta \beta^{(L)}} \frac{\delta a^{(L)}}{\delta z^{(L)}} \frac{\delta J^{(L)}}{\delta a^{(L)}}$$

⇒ error of the output layer

$a^{(L-1)}$ connection between the layers. Each layer is only dependent on the previous layer

Derivative of sigmoid.

it depends on which loss function we use.
$a \rightarrow h_\theta(x) \rightarrow g(z)$

☆ Advantages and Disadvantages

优:
① strong information on the entire network
② Ability to work with incomplete knowledge
③ Having ~~de~~fault tolerance
④ Having a distributed memory
⑤ Gradual corruption.
⑥ parallel processing capability

缺:
① Finding the right architecture is hard because it's an art
② The final model is a black box that is hard to explain.
③ It needs lots of (tens of thousands, millions) data.

CNN (卷积神经网络)

CNN is popular when solving computer vision problems.

What is convolution in 1-D

$$(f*g)(x) = \int_{-\infty}^{\infty} f(k)\, g(x-k)\, dk$$

☆ The convolution is the overlap between f and g

Challenges with NN:

1. lots of data
2. overfitting
3. ~~A~~ Hyperparameter tuning
4. ~~Requa~~ Requires high-performance hardware
5. blackbox (hard to explain)

Why NN popular

1. Cheap computation (GPU)
2. availability of Large datasets
3. Versatile in NLP, Speech identification, Computer vision ...

NN are run on GPU's and NVIDIA is the leading GPU contractor.
TensorFlow and Cafe are the ~~bes~~ best for NNs.

贝叶斯

Classical

- Sample a population n times
- construct a relationship between feature and outcome based on the distribution you discover in your sample.
- Apply this relationship to make predictions or infer things about causality.

Bayesian.

- Develop a distribution for your feature based on all knowledge you have at the time — Prior Distribution.
- sample a feature
- Use the feature and the prior distribution to create a new distribution that reflects all of your old knowledge plus the new bit of information. — Posterior Distribution

优点：
① Utilizes prior knowledge
② Best way to model low occurance events
③ Good way to model problems that are hard to sample
④ Good way to model problems that are dynamic
⑤ Great way to come at a problem from a different direction.

$$P(\Theta|D) = \frac{P(D|\Theta) * P(\Theta)}{\underbrace{\sum_{k=1}^{m} P(D|\Theta_k) * P(\Theta_k)}} \rightarrow \text{normalization factor}$$

Terminology:

$P(\Theta)$ is the prior distribution for Θ （也就是Y）

$P(D|\Theta)$ is the likelihood that you got this observation given that prior knowledge.

$P(\Theta|D)$ is your posterior

这里的内容 LDA 那里有更详细的解释.

Discrete priors

$$P(\Theta|D) = \frac{P(D|\Theta) * P(\Theta)}{\sum_{k=1}^{m} P(D|\Theta_k) * P(\Theta_k)}$$

Continuous priors

$$P(\Theta|D) = \frac{P(D|\Theta) * P(\Theta)}{\int P(D|\Theta) * P(\Theta) * d\theta}$$

Sometimes, it's hard to find the likelihood. So, what ~~to~~ should we do to solve for posterior?

① Conjugate function
② MCMC
③ BBN.

Ex. flip a coin $P(H)=\theta$, $P(T)=1-\theta$

So, the probability of getting a particular sequence of flips
$\Rightarrow P(D) = \theta^h \cdot (1-\theta)^t \Rightarrow P(D|\theta)$

Suppose we don't know if the coin is unbiased.
Assume the prior $P(\theta) = 0.25, 0.5, 0.75$.
which means tail biased, unbiased, head biased.
So, we have the likelihood and prior, we can compute the posterior using them

☆ So, the prior function and likelihood function are conjugate functions if the resulting posterior function is of the same form as the prior.

So, back to coin problem: We know likelihood is

$$P(D|\theta) = \theta^h (1-\theta)^t \quad (\text{No flexibility})$$

But we are free to pick the prior model $(P(\theta))$ as long as it represents the belief.

Beta function is conjugate with the coin toss likelihood function

$$P(\theta|a,b) = \frac{\theta^{(a-1)}(1-\theta)^{(b-1)}}{B(a,b)} \rightarrow \text{normalization factor}$$

$$B(a,b) = \int_0^1 d\theta \ \theta^{(a-1)}(1-\theta)^{(b-1)}$$

So, beta function depends on a and b, that determine the shape.
We use the beta distribution as the prior for Bayes ~~eq~~ and combine it with the likelihood for the coin problem

$$P(\theta \mid a,b,N,z) = \frac{\theta^{(a-1+z)}(1-\theta)^{(b-1+N-z)}}{B(a+z,\, b+N-z)}$$

where ~~P(θ)~~ $p(D \mid \theta) = \theta^{z}(1-\theta)^{(N-z)}$

There is no need to solve a complicated Integral for the posterior since the numerator of the posterior is just another beta distribution

$$P(\theta \mid z,N) = \frac{P(z,N \mid \theta)\, P(\theta)}{P(z,N)}$$

$$= \frac{\text{Bernoulli} * \text{beta}(a,b)}{\text{Normalization}}$$

$$= \frac{\theta^{z}(1-\theta)^{(N-z)} * \theta^{(a-1)}(1-\theta)^{(b-1)}}{B(a+z, b+N-z) * P(z,N)}$$

$$= \text{beta}(a+z,\, b+N-z)$$

So, if the prior distribution is a beta(a,b) and the data gives you z heads and N flips, the posterior is beta(z+a, N-z+b)

Properties of Beta Distribution.

$$\bar{\theta} = a/(a+b)$$

$$\bar{\theta}_{posterior} = \frac{z+a}{z+a+N-z+b}$$

$$= \frac{z+a}{a+N+b} = \frac{a}{a+b} \cdot \frac{a+b}{N+a+b} + \frac{z}{N} \cdot \frac{N}{N+a+b}$$

$$= \text{prior average} * \text{weight} + \text{Data proportion} * \text{weight}$$